CIT 591 Introduction to Software Development

Week 3: Introduction to Code Design

Module Learning Objectives

- Analyze another programmer's code in terms of design
- Write interrelated code in the same class
- Separate responsibilities from the actual implementation details
- Minimize copying and pasting code

Module Glossary

- **DRY (Don't Repeat Yourself):** A design principle that says that copy pasting the exact same code into several different places is not desirable.
- **Principle of least surprise:** The design of a method/class should match the user's expectation.
- **Coupling:** The measure of the degree of interdependence between the various classes in your project. A good project will have low coupling.
- **Cohesion:** The degree to which the methods inside a class belong together. A good designer will ensure their class has a high amount of cohesion.

Module Resources

- **Recommended Textbook Readings:**
  - Designing Classes & Methods with CRC Cards:
    - 12.1: Classes and Their Responsibilities
  - Design Principles:
    - 8.1: Discovering Classes
    - 8.2: Designing Good Methods
  - Preparing for this Week's Recitation and Assignment 2:
    - 6.9 Application: Random Numbers and Simulations
- **Websites:**
  - CRC Cards
  - Kent Beck, Ward Cunningham - A laboratory for teaching object oriented thinking
  - Examples of Monte Carlo Simulations

Penn Engineering
ONLINE LEARNING

Key Concepts & Examples

CRCs: Before you write a single line of code, it's a good idea to think through what attributes and actions your classes will need, as well as what other classes they will need to interact with.

A useful design practice to accomplish this is using a physical or digital CRC card, which stands for Class, Responsibilities, Collaborators. Doing this with physical cards makes it easier to arrange your cards to see collaboration between classes more easily.

- Examples used in the video: **student** CRC example (students have a name and a GPA, can ask the teacher questions, can study; student class collaborates with course class and teacher class); **soccer game** CRC example (classes include a team, a referee and a player).

Design Principles: DRY (Don't Repeat Yourself), Principle of Least Surprise, Coupling, Cohesion. Following these design principles will make your code more robust and easier to write, no matter what programming language you're using.

It's recommended to consider principles like low coupling and high cohesion early on in your project, at the design stage before you have written much (if any) code. It will make you a faster and more efficient programmer, as you'll learn to reuse code and your code will be less error-prone.

- Examples used in the video: Expected output of an `isEven` method (principle of least surprise); continuing the soccer player/ref example for recommending low coupling. As a counter example of coupling that is too high, we gave the example of a student class that includes a bankAccount class.